Journal of Network Communications and Emerging Technologies (JNCET)
Volume 10, Issue 2, February (2020)

Intent-Based, Voice-Assisted, Self-Healing SDN Framework

Mansi Jain [1], Shikha Suneja [2], Srinidhi Vajapeyam Srivatsa [3], Vaishali Ananthasubramanian [4], Yogisai Maramraj [5], Levi Perigo [6], Rahil Gandotra [7], Dewang Gedia [8]
[1, 2, 3, 4, 5, 7, 8] Interdisciplinary Telecom Program, University of Colorado Boulder.
[6] Department of Computer Science, University of Colorado Boulder.
mansi.jain@colorado.edu [1], shikha.suneja@colorado.edu [2], s.vajapeyamsrivatsa@colorado.edu [3], vaishali.ananthasubramanian@colorado.edu [4], yogisai.maramraj@colorado.edu [5], levi.perigo@colorado.edu [6], rahil.gandotra@colorado.edu [7], dewang.gedia@colorado.edu [8]

Published online: 29 February 2020

Abstract – The rise in cloud-based service offerings has increased the scale and complexity of networks. Previous research indicates that network management tasks using the command-line interface (CLI) and primitive scripting do not scale, as they are complicated, slow, and inefficient. In this research, a software-defined networking (SDN) framework is developed to help solve these problems. Using the intent-based, voice-assisted, self-healing network framework, a proof of concept tool is developed which can make these tasks simple, fast, and efficient by providing an abstraction layer to the operator. The results of this research indicate that, by leveraging the concepts of SDN, it is possible to build a robust and scalable solution that provides the visibility and control needed to effectively achieve network administration, troubleshooting, and self-healing tasks by issuing verbal intents through a digital voice-assistant.

Index Terms – Amazon Alexa, BGP, Intent-Based Networking, Network Management, OpenFlow, Python, Quality of Service, SDN, Self-Healing.

1. INTRODUCTION

As networks have grown in scale, it has become arduous for network operators to make accurate decisions while managing and troubleshooting networks [1]. While network automation has eased the tasks of manual configuration, network operators must still make network management decisions manually after obtaining and analyzing information about the network. Keeping business interests in mind, they must then make a rational decision about how to manage the network under those circumstances. As the network and application protocol stacks have grown with additional layers of system virtualization, it has also become harder for engineers to isolate network issues in real-time, at a large scale [2, 26, 28, 29].

A network operator has to determine 'what' to do, 'how' to do it, and 'when' to do it. However, it is not pragmatic to make these decisions quickly on a large scale [3]. Using an intent-based network (IBN) can simplify this process. The concept of an intent is that the network operator only communicates 'what' needs to be done and the system will use its intelligence to determine 'how' best to do it, by leveraging real-time network information [4].

A major setback in designing intelligent networks is that network devices only have a local perception of the network state, which hinders their ability to effectively determine the 'how' and 'when' aspects of making networking decisions [5]. One of the concepts of software-defined networking (SDN) is to treat the network as one entity instead of a set of individual boxes because an SDN controller is a logically centralized entity with an end-to-end view of the network [29, 6]. The research presented in this paper proposes an intent-based framework to delegate some tasks of the network administrator to the SDN controller so they can be performed more efficiently. For example, the SDN controller can be programmed to collect and audit real-time network information, and process this information to make a rational decision, which is in purview of the intent framework set by the network administrator. Since the tasks that were previously performed by the network administrator have now been delegated to software, it is possible to scale the system and process a larger set of information faster. Utilizing this, the network is administered through a set of software applications, when the network administrator issues an intent to the system. This makes it possible to use the concepts of DevOps to perform network administration, network management, and troubleshooting tasks more effectively, and in a structured and consistent manner even across a large network [27, 7]. Furthermore, the framework provides a voice assistant to facilitate the network administrator's interaction with the system and provides another layer of abstraction to the network management and operations. The motivation behind this work was to reduce the management and troubleshooting overheads in large networks by developing a framework that abstracts and automates the resolution steps. The remainder of the paper is organized as follows: Section 2 reviews existing work and briefs the novelty of our work, Section 3 describes the research methodology including the environment and technologies incorporated. Section IV provides the research results and

analysis, and Section V concludes our research and addresses the scope for future enhancements.

2. RELATED WORK

2.1. Voice assistants

As the popularity of the Internet of Things (IoT) has increased, there has been a rise in devices that connect to wireless networks and communicate over the Internet [8]. In addition, many of these IoT devices include a voice service, known as cloud-based intelligent virtual assistant (IVA) [9]. Milhorat et al. [10] stated that the voice-based IVA fulfils the most requested requirements of a personal assistant – simplicity, flexibility and easiness of interaction - and with voice-based IVAs, the input and output interfaces require less cognitive function and attention from the users.

Rajalakshmi and Shahnasser [11] proposed a solution to integrate IoT devices to voice-based IVAs and to control those using Amazon Web Services (AWS) offerings like AWS IoT, AWS Lambda, AWS EC2 and Alexa skill developer. They used a Raspberry Pi along with plugins to configure networking between all the devices in their simulation, and they were successful in using the Raspberry Pi to control the devices by issuing voice intents through Amazon Alexa. This research demonstrated a method to control IoT devices through voice commands and signified a paradigm shift of new technologies moving towards a voice-based front-end.

2.2. Intent-based SDN

Chaudhari et al. [12] demonstrated a visually represented, intent-based, voice-assisted networking system (VIVoNet). They employed Amazon Echo as the voice assistant to convey intents that configured the SDN-based infrastructure. Their research proposed an IBN with a voice interface that could abstract the underlying network infrastructure and allow the administrators to alter its behavior by merely expressing their intents via voice commands. VIVoNet implemented the following intents:

- Least Latency: Configuration of a path with minimum delay from source to destination.
- High Bandwidth: Configuration of a path with maximum available bandwidth between source and destination.
- Least Hop Count: Configuration of the shortest path between source and destination, based on a hop count metric.

VIVoNet used a static topology file to reference all the network devices that they used, and their work focused on developing applications that could configure the network. When a user intent was issued, the system would use the topology file as a reference to push appropriate network configurations. Since a static mapping of the network needed to be maintained in a topology file, the implementation had scaling challenges.

The work in this paper builds on this idea to develop an SDN framework equipped with a suite of applications that will enable network administrators to deliver network administration, network management and troubleshooting tasks by simply issuing voice intents through a digital voice assistant. In addition, the framework is developed to work with a dynamic topology, thus adding flexibility, reliability and scalability to our system.

2.3. Network monitoring with SDN

Fault detection and troubleshooting is an integral part of a network administrator's work. Therefore, there could be a significant benefit in being able to perform these tasks in a fast and error-free manner. Xie et al. [13] found that between 2007 and 2013, around 28 cloud providers amassed total losses of up to U.S. $273 M and incurred a loss of 1,600 hours due to disruptions caused by application and infrastructure failures.

Haque and Moyeen [14] defined the difference between network faults and failures - failures are mostly link/node failures and faults are mainly software bugs or hardware malfunctions. However, their research overlooked a scenario where multiple devices could generate multiple fault alarms, causing inconsistencies and potential delays in network convergence after a failure [15]. In addition, some types of failures, such as operator errors and misconfigurations are harder to detect and troubleshoot, because network administrators are constrained to using ad-hoc tools. Even though the distributed nature of traditional networks can provide scalability and resiliency, it hampers the management of complex network environments because of the larger scope involved during troubleshooting [17].

Identifying the benefit of SDN over traditional networks, Atary and Bremler-Barr [16] conducted an experiment to monitor all links in a static topology that they created, called Granular RTT Monitoring Infrastructure (GRAMI). They also detected the round-trip path between any two switches in the network using the concepts of SDN. They were able to accomplish this by using the southbound protocol OpenFlow and installing two new flows each time a request was made. While the research was able to perform monitoring on an SDN with a high-level of granularity, it was dependent on preconfigured proactive flows on the switches added for monitoring.

The work in this paper proposes a scalable framework, which can dynamically monitor network information across any network, without introducing the complexity for the network administrator to manually determine, add and manage proactive flows.

2.4. Troubleshooting and self-healing

Even though the concepts of self-healing are extensively used in other industries, self-healing in the field of networking is still in its nascent phase and experimentation to use it extensively is being administered by companies such as CenturyLink with

SHNS platform [18], Facebook with their FBAR platform [19], LinkedIn with Nurse platform [20], and Netflix with their Winston tool [21]. A self-healing topology discovery (SHTP) application was designed by Ochoa-Adal et al., which could discover the network topology [22]. The application was able to accomplish this without needing any static information such as a preconfigured IP address. Instead, the research decoupled static dependencies to increase resiliency of the network. The research also added a self-healing feature, which kept the network up by detecting and resolving port down status and connectivity issues in network devices.

With the proposed work in this paper, the goal is to enhance the body of knowledge to develop a consistent framework for network troubleshooting. For example, a port down status could indicate a healthy network state if that port was intended to be down by the network administrator. Connectivity issues relating to a network device is expected if the device is under maintenance. This work, unlike previous research, does not rely on primitive triggers such as port down or connectivity down to flag network issues, but instead, it uses a Network Source of Truth (NSoT) file as a reference for detecting and flagging misconfigurations. The NSoT contains absolute truth about the intended state of the network, which prevents the proposed framework from reacting to false positives. The self-healing application implemented in this research focuses on automatically detecting and resolving network issues related to OpenFlow and Border Gateway Protocol (BGP).

2.5. Research novelty

The previous research provides dispersed solutions for configuring networks or performing network monitoring and self-healing that are topology specific or only work on static topologies with preconfigured flows. These solutions lack structure and consistency, which make it hard to incorporate them on dynamic or large network environments. There is minimal research that provides a consistent and scalable framework for performing the tasks of network administration, network management, and troubleshooting programmatically by issuing intents through a voice assistant.

The primary research question answered via this research was can a framework be developed for performing network administration and troubleshooting by issuing voice intents. We identified and answered the following sub-problems to effectively develop and evaluate the research.

a) Can voice intents be used to dynamically gather network information?

b) Can network management tasks be performed by issuing high-level voice intents?

c) Can network troubleshooting and self-healing be performed by issuing high-level voice intents?

d) Can security policing be performed by issuing high-level voice intents?

3. SYSTEM SETUP

The functional diagram of the proposed framework (see Figure 1) illustrates the components of the system. A Ryu SDN controller is used to decouple control plane functionalities from the SDN infrastructure [23]. The user interface of the framework handles interaction between the user and the system. It includes a voice assistant and a web graphical user interface (GUI). The voice assistant processes the user intent as an input and communicates the results back to the user. The web-based GUI is used to render visual results such as displaying the topology.

The SDN infrastructure consists of Open vSwitches (OvS) which are interconnected to form a Clos network [24]. The SDN infrastructure and the controller are virtualized using VMware ESXi - enterprise-class, type-1 hypervisor. The VMware vCenter client is used to manage the virtualized hosts. The hypervisor is hosted on a Dell PowerEdge R420 server, which consists of 12 core CPUs, 32 GB of RAM, and a local storage of 400 GB. The Virtual Machine (VM) hosting the SDN controller is allocated 4 GB of RAM and is loaded with an Ubuntu 16.04 Operating System (OS). All other VMs hosting OvS are allocated 2GB of RAM and loaded with an Ubuntu 16.04 OS. The OvS version 2.0.2 Linux package is installed on the VMs.

The SDN controller, interacts with the SDN infrastructure using OpenFlow v1.3 as the southbound protocol. This interaction can be read/write, wherein the controller can either monitor network-related information or write hardware flows into the forwarding plane of these devices. The network administrator (user) interacts with the voice assistant to convey an intent via voice input. The intent is processed by the voice-assistant and if recognized, will trigger the corresponding networking application to act. A set of Python-based network applications run on the Ryu SDN controller, which perform the tasks of network monitoring, network management, and troubleshooting based on the intents that are provided.

In this research, the following applications are implemented with the proposed framework to answer the sub-problems and research question stated previously:

3.1. Topology Discovery

By issuing an intent, a network administrator can obtain real-time network information. For example, if the intent is to "visualize the topology," the system gathers real-time topology information and visualizes the complete network topology at that instant, on a front-end GUI.

3.2. Quality of Service (QoS) Management:

Intents can be issued to configure and migrate QoS queue resources based on network parameters such as packet drops or link utilization. The network administrator could also have the system continuously monitor network parameters and

reconfigure QoS queue settings as appropriate, in real-time, to counter congestion.

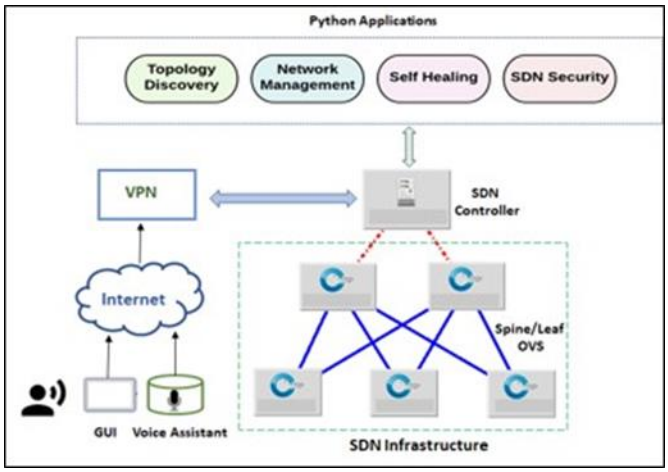

Figure 1 Functional Diagram

3.3. BGP Self-Healing

In a scenario where network connectivity is down, network administrators can issue intents to detect network faults. In addition, the system can be instructed to rectify any network anomalies and rollback the network to a healthy state. With self-healing, the system detects and corrects issues automatically. For example: detecting a BGP neighbor down state and correcting it.

3.4. SDN Security

By issuing a high-level network intent, the network administrator can secure the entire SDN by applying a centralized policy. For example, the network administrator can issue an intent to flag a Denial of Service (DoS) attack beyond a set threshold and take preemptive action to secure the network.

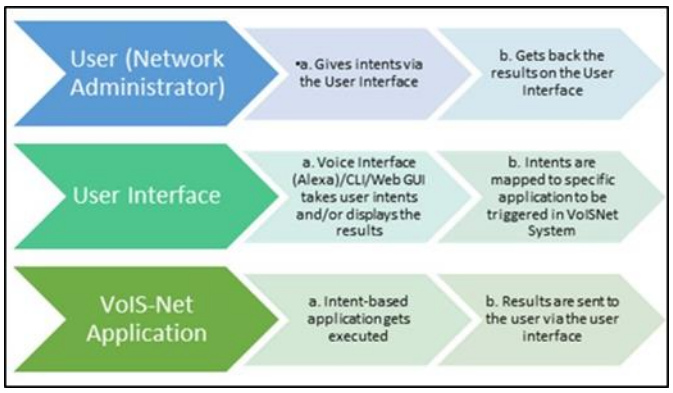

Figure 2 User–Application Interaction

The input, processing, and output mechanism of these applications are described in Figure 2. When a user intent is provided as an input, the Python-based applications are called to perform one or more tasks in response to the intent. Upon performing these tasks, the applications provide a response back to the user.

4. RESULTS AND ANALYSIS

The functional diagram of the proposed framework (see Figure 1) illustrates the components. By developing the following applications and testing them, the researchers attempted to find answers to the three sub-problems and the research question stated previously.

The framework uses Amazon's Alexa skill developer service for developing a voice interface where network administrators can issue intents to perform networking tasks. To accomplish this, a Python3 development environment was setup and Ngrok and Flask-ask packages were used along with the Alexa skill developer to process voice intents. Custom skills are added on the Alexa skill developer application to recognize specific intents related to this research. These custom skills are hosted on a public HTTPS endpoint provided by Ngrok.

When Alexa receives a voice command, it is processed at this endpoint and matched with the database of all skills defined in the Alexa skill developer module. If this command is recognized to belong to an intent implemented by the framework, one or more specific actions are taken accordingly as discussed in the following section.

4.1 Can Network Information be Gathered Dynamically by Issuing High-Level Voice Intents?

When the user says to Alexa: "Visualize the topology," the voice command is received and processed by the HTTPS endpoint, where it compares the voice command with a wide range of intents defined in the Alexa skill developer module. In this case, the voice command is recognized to belong to an intent defined in our framework, which is to gather and visualize topology information, and therefore it invokes the Topology Discovery application to take the following set of actions, as explained below.

When the Topology Discovery application is invoked, it discovers network connections and link status information in real-time. This information is stored in a database as shown in Figure 3. Furthermore, the application dynamically tracks the network, based on OpenFlow messages and updates the topology information whenever there is a change in the network. The Ryu topology API is leveraged to collect this information on an event basis. The following OpenFlow events will trigger a change in topology information:

- Switch Enter: This event is triggered when a switch successfully establishes an OpenFlow session with the SDN controller. The switch is then assumed to 'enter' the topology.
- Switch Leave: This event is triggered when a switch tears down an existing OpenFlow session with the SDN

controller. The switch is then assumed to 'exit' the topology.

- Port Modified: This event is triggered whenever there is a change in the status of an OpenFlow port on a switch.

The information from the topology database was also used to render the user with a visual representation of the topology as shown in Figure 4 and Figure 5, for different test cases. Based on the results, the following observations are made relating to the development of the Topology Discovery application using the framework, which answer this sub problem:

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: topo_connections

	source_dpid	dest_dpid	source_port	dest_port	status
	Filter	Filter	Filter	Filter	Filter
1	52235376589	52238047428	4	5	UP
2	52239256007	52238047428	4	6	UP
3	52229058563	52238047428	4	4	UP
4	52230270179	52229058563	6	3	UP
5	52230270179	52239256007	4	3	UP
6	52230270179	52235376589	5	3	UP

Figure 3 Link Information in Topology

i. It is possible to dynamically obtain an end-to-end view of the network by issuing a voice intent. This provides a high-level of abstraction and is different from previous research work where the network administrator needed to either preconfigure flows or maintain a static topology file to perform similar tasks.

ii. Similar applications can be developed using the proposed framework to collect miscellaneous information using OpenFlow messages in real-time. For example, by collecting information of available link bandwidth, it is possible to determine the best path between any two endpoints in the network at that instant. Since the information collection is dynamic and not dependent on any specific topology, the proposed framework provides a consistent and scalable methodology to collect information in a network where network devices support OpenFlow v1.3.

4.2 Can Network Management Tasks be Performed by Issuing High-Level Voice Intents**?**

By implementing the QoS Management application, an attempt is made to identify if network management tasks can be performed by issuing voice intents. The goal of the QoS Management application is to achieve better utilization of network resources by dynamically adjusting the bandwidth window for every queue that is configured. This is done by continuously tracking queue-specific information such as the volume of traffic and the number of packets dropped for traffic matching that specific queue.

The network administrator can issue intents to configure and manage these queues. A user intent can be issued to pre-set a threshold value of packet drops after which the bandwidth window must be adjusted and the increment/decrement value of bandwidth. For example, if a threshold of 0.6% is set with an increment/decrement factor of 2 Mbps, whenever the percentage of packet drops/transmission errors (TX_ERRORS) per queue exceed 0.6, the bandwidth window of that queue is re-adjusted to increase the bandwidth allocation by 2 Mbps for that queue. In developing the QoS Management application, the following Ryu SDN controller applications were used to complement the application: rest_qos, simple_switch_13, rest_conf_switch and ofctl_rest. To register flows related to configured queues, the simple_switch_13 application had to be modified to add a flow entry with an OpenFlow Go-To action.

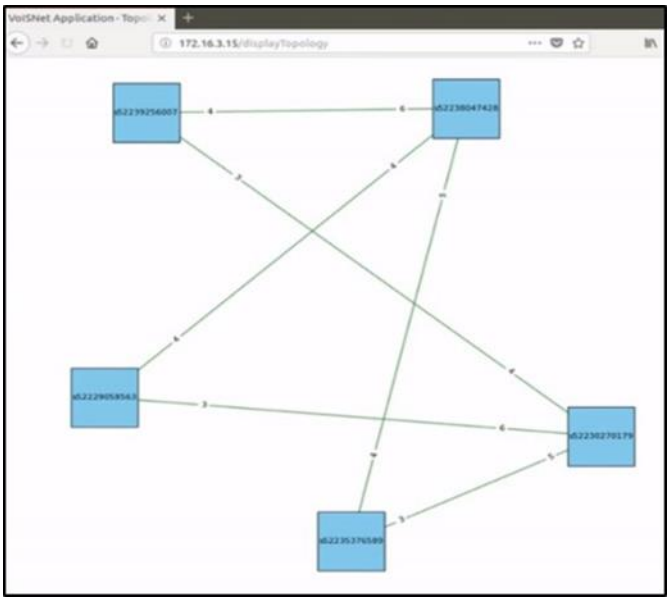

Figure 4 GUI Output of Healthy State

The QoS Management application was tested in a network with preconfigured queues configured to match protocol-specific traffic. There were two queues configured, one that would match Transmission Control Protocol (TCP) traffic and another that would match User Datagram Protocol (UDP) traffic. The testing of the QoS Management application was administered in the following stages:

i. Ability to accept user input: The QoS Management application runs continuously in the background. However, at any point, the user must be able to issue intents to configure queue management settings such as setting threshold of packet drops and increment/decrement value of bandwidth as discussed before.

ii. Simulation of access traffic: Using iPerf, simulation of access traffic was verified to check if traffic could be generated with bandwidth attribute for UDP and window size for TCP to carry out further test cases.

iii. Monitoring packet drops: The application was tested to verify that it could detect packet drops on specific queues in real-time, by monitoring OpenFlow messages.

iv. Dynamic reconfiguration of queue resources: Test cases were devised to verify when packet drops for a queue exceeded a threshold, the application would reconfigure queue resources by increasing the bandwidth allocated to that queue. REST calls were made to retrieve queue statistics before and after triggering packet drops to exceed the set threshold.

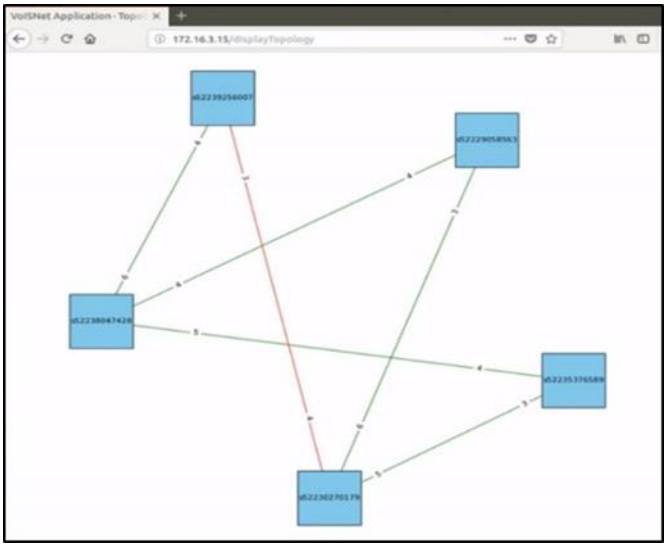

Figure 5 GUI Output of Link Down

Based on the results, the following observations are made relating to the development of the QoS Management application using the proposed framework, which answer this sub problem:

i. Many tasks of network management are cumbersome and time consuming, especially in large and complicated networks. By implementing QoS Management within our framework, the ease of using voice intents to configure and manage QoS queue settings is illustrated.

ii. Similar applications can be developed using the proposed framework to perform various network management tasks by providing a high level of user abstraction.

4.3 Can network troubleshooting and self-healing be performed by issuing high-level voice intents?

By implementing the BGP Self-healing application, an attempt is made to identify if network troubleshooting and self-healing can be performed by issuing voice intents. By leveraging the framework developed, the application will enable building a self-reliant system that can detect network faults and resolve them as per the intent of the network administrator. With the BGP Self-healing application, the focus is to detect, and correct network issues related to OpenFlow and BGP configuration, to achieve end-to-end connectivity between two hosts across the network. We can largely classify the application to consist of the following two features:

i. Event/Fault-Detection:

Upon issuing an intent to "detect network issues" or "detect and resolve network issues," the application gathers the current network state and configuration information. It will compare current configuration with the expected or last working configuration referenced in the NSoT file. The application will then flag any misconfigurations and/or unexpected network states.

In testing, several combinations of test cases were run by injecting errors in the OpenFlow and BGP configuration, such as misconfiguration of controller IP, OpenFlow version, BGP peer IP, and remote AS number. In each case, upon issuing a verbal intent to "detect network issues," the application was able to detect and validate misconfigurations present on network devices by comparing it with the NSoT file. The detected errors were logged into a file and reported back to the user through the digital assistant.

ii. Fault-Resolution/Self-healing:

Upon detecting misconfigurations and/or unexpected network states when compared with the NSoT file, the system will initiate a rollback to the last working configuration using the NSoT file as the reference. The hook for this application can either be an explicit intent issued by the network administrator to resolve specific network issues identified previously using the detection application. Alternatively, the hook can be a pre-set intent to the system to "continuously detect and resolve network issues," which would correspond to the self-healing feature set of this framework.

In testing, several combinations of test cases were run by injecting errors in the OpenFlow and BGP configuration, such as misconfiguration of controller IP, OpenFlow version, BGP peer IP, and remote AS number. With these misconfigurations in place, the end hosts did not have reachability to each other, because the switches did not have an active OpenFlow connection with the controller and the BGP session was inactive. A verbal intent was issued to "detect and resolve network issues" upon which the system was able to identify misconfigurations and rollback the configuration to be consistent with the NSoT file. At this point, the end hosts were able to reach each other owing to active OpenFlow and BGP sessions that were reestablished.

Based on the results, the following observation was made relating to the development of the BGP Self-healing application using the proposed framework, which answer this sub problem:

i. It is possible to perform troubleshooting and self-healing in a network by issuing high-level voice intents. The application is able to detect and correct network issues

dynamically when a network administrator issues an intent to "detect and resolve network issues." In cases where the application fails to self-heal, it will log issues detected and steps taken by the system in its attempt to resolve network issues. The network administrator can refer to the log report and then continue troubleshooting larger issues. Therefore, even when self-healing does not resolve the issue, it will save time by eliminating the need for the network administrator to manually carry out each step of troubleshooting.

4.4 Can security policing be performed by issuing high-level voice intents?

By implementing the SDN Security application, an attempt is made to determine if security policing in a network can be performed by issuing voice intents. The concepts of SDN allows for implementing centrally enforceable security policies. However, it also exposes additional attack vectors, which could aim to take down the SDN controller by carrying out a control plane DoS attack. As part of this research, a Python-based application was developed to detect and mitigate variations of DoS attacks on the SDN controller at run-time. The implementation involves monitoring the incoming traffic (OpenFlow PACKET_IN messages) on the SDN controller and alarming an attack if the number of PACKET_IN messages reaches a set threshold (for instance, 100 packets). The attack is then mitigated by adding security rules to block malicious traffic from the identified attacker. The network administrator can set this threshold and alter it at any time by simply issuing a voice intent to Alexa.

Testing was administered in the following steps:

i. Ability to accept intents: The SDN Security application runs continuously in the background. However, at any point the user must be able to issue intents to enforce security policing. For example, the security policy may involve the user setting a threshold for number of PACKET_IN messages within a period of time beyond which it should be considered as a DoS attack. The user may enforce these policies at any time. For instance, after noticing certain traffic patterns, the network administrator may change these settings by issuing a voice intent.

ii. Ability to detect attacks and act: A DoS attack was simulated by generating a large number of PACKET_IN messages sent to the controller. The SDN Security application was able to detect an attack upon reaching the threshold value of PACKET_IN messages. The application then successfully took preemptive action to block and log further PACKET_IN messages originating from the IP address of the attacker. This was verified by checking traffic on the wire and verifying packet drops logged in the iptables rule that was added. After exceeding the threshold, all PACKET_IN messages originating from the IP address of the attacker were dropped.

Based on the results, the following observation was made relating to the development of the SDN Security application using the proposed framework, which answers this sub problem:

i. Using the framework developed to implement security in an SDN, the ease of using voice intents to enforce security policies was illustrated.

5. CONCLUSION

Previous work provided dispersed solutions, which lacked a consistent and structured approach to programmatically perform tasks of network administration, network management and troubleshooting. Notably, this made performing these tasks harder and time consuming for network administrators. Previous solutions only worked on static network environments, which introduced scalability challenges.

This research introduces a robust, intent-based framework for performing tasks of network administration, network management and troubleshooting in SDN by issuing voice intents. A suite of SDN applications were developed using the proposed framework as proof of concepts to illustrate the inherent advantages by using this framework. It was found that applications built are easy to use and provide a high level of abstraction for the user. In addition, the proposed framework makes it possible to develop applications that work dynamically with any network environment, provided the network devices support OpenFlow v1.3. This adds flexibility, reliability, and scalability into the system.

5.1. Future Work

This research presented a proof of concept for developing applications using the Intent-based, Voice-assisted, Self-healing SDN framework. The framework provided a consistent methodology to develop applications, which could perform tasks of a network administrator based on voice intents. The following are some additional feature sets that could be included:

- Incorporating the concepts of machine learning to more efficiently perform tasks of network troubleshooting, self-healing, and security policing.
- Incorporating Proactive APIs for Alexa where the voice-assistant can provide unsolicited insights to the network administrator that could help optimize network administration and network management.

REFERENCES

[1] Internet World Stats. World Internet Users Statistics and 2019 World Population Stats. Retrieved September 23, 2019 from https://www.internetworldstats.com/stats.htm

[2] CA Technologies. 2018. The Critical Guide to Modern Network Monitoring. https://www.ca.com/content/dam/ca/us/files/ebook/the-critical-guide-to-modern-network-monitoring.pdf.

[3] W. C. Goers and M. R. Brenner, "Implementing a Management System Architecture Framework." Bell Labs Technical Journal, vol. 5, no. 4, pp. 31–43, 2002.
[4] T. Szyrkowiec et al., "Automatic Intent-Based Secure Service Creation Through a Multilayer SDN Network Orchestration," Journal of Optical Communications and Networking, vol. 10, no. 4, 2018.
[5] T. Keary. 2018. What is Software Defined Networking (SDN) and why is it important? Retrieved September 23, 2019 from https://www.comparitech.com/net-admin/software-defined-networking/.
[6] K. Raghunath and P. Krishnan, "Towards A Secure SDN Architecture," 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2018.
[7] E. Madison. 2017. NetDevOps: what does it even mean? (October 2017). https://cumulusnetworks.com/blog/netdevops-meaning.
[8] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges," 10th International Conference on Frontiers of Information Technology, 2012.
[9] H. Chung, J. Park, and S. Lee, "Digital forensic approaches for Amazon Alexa ecosystem," Digital Investigation, vol. 22, 2017.
[10] P. Milhorat, S. Schlogl, G. Chollet, J. Boudy, A. Esposito, and G. Pelosi, "Building the next generation of personal digital Assistants," 1st International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), 2014.
[11] A. Rajalakshmi and H. Shahnasser, "Internet of Things using Node-Red and alexa," 17th International Symposium on Communications and Information Technologies (ISCIT), 2017.
[12] A. Chaudhari et al., "VIVoNet: Visually-Represented, Intent-Based, Voice-Assisted Networking," International Journal of Computer Networks & Communications (IJCNC), vol. 11, no. 2, Mar. 2019.
[13] J. Xie et al., "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research Issues and Challenges," IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 393-430, 2019.
[14] A. I. Haque and M.A. Moyeen, "Revive: A Reliable Software Defined Data Plane Failure Recovery Scheme," Network and Service Management (CNSM) 14th International Conference, pp. 268–274, 2018.
[15] Y.-Z. Liao and S.-C. Tsai, "Fast Failover with Hierarchical Disjoint Paths in SDN," IEEE Global Communications Conference (GLOBECOM), 2018.
[16] A. Atary and A. Bremler-Barr, "Efficient Round-Trip Time monitoring in OpenFlow networks," IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, 2016.
[17] J. Sargent. 2018. SDN makes troubleshooting network issues painful. Retrieved September 25, 2019 from http://www.itopstimes.com/itops/sdn-makes-troubleshooting-network-issues-painful/.
[18] CenturyLink. 2017. Self-healing Network Service (SHNS). Retrieved September 25, 2019 from http://www.centurylink.com/wholesale/pcat/selfhealingntwksvcs.html.
[19] Facebook. 2011. Making Facebook Self-Healing. Retrieved September 25, 2019 from https://www.facebook.com/notes/facebook-engineering/making-facebook-self-healing/10150275248698920.
[20] B.C. Sherwin. 2015. Introducing Nurse: Auto-Remediation at LinkedIn. Retrieved September 25, 2019 from https://engineering.linkedin.com/sre/introducing-nurse-auto-remediation-linkedin.
[21] Netflix. 2016. Introducing Winston — Event driven Diagnostic and Remediation Platform. Retrieved September 25, 2019 from https://medium.com/netflix-techblog/introducing-winston-event-driven-diagnostic-and-remediation-platform-46ce39aa81cc.
[22] L. Ochoa-Aday, C. Cervello-Pastor, and A. Fernandez-Fernandez, "Self-Healing Topology Discovery Protocol for Software-Defined Networks," IEEE Communications Letters, vol. 22, no. 5, pp. 1070-1073, 2018.
[23] Ryu. Build SDN Agilely. Retrieved September 26, 2019 from https://osrg.github.io/ryu/.
[24] Open vSwitch. Retrieved September 26, 2019 from https://docs.openvswitch.org/.
[25] R. Gandotra and L. Perigo. "SDNMA: A Software-Defined, Dynamic Network Manipulation Application to Enhance BGP Functionality," 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1007-1014, Jun. 2018.
[26] D. Gedia and L. Perigo, 'A Centralized Network Management Application for Academia and Small Business Networks,' Information Technology in Industry Journal, vol. 6, no. 3, pp.1 – 10, 2018.
[27] D. Gedia and L. Perigo, 'Performance Evaluation of SDN-VNF in Virtual Machine and Container' in IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp. 1-7, 2018.
[28] D. Gedia and L. Perigo, 'Latency-Aware, Static, and Dynamic Decision-Tree Placement Algorithm for Containerized SDN-VNF in OpenFlow Architectures' in IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Dallas, USA, 2019.
[29] A. Jain et al., "Trend-Based Networking Driven by Big Data Telemetry for SDN and Traditional Networks," International Journal of Next-Generation Networks (IJNGN), vol. 11, no. 1, 2019.

Authors

Mansi Jain is a graduate student from the University of Colorado Boulder with a major in Network Engineering. She received her Bachelor of Engineering in Electrical and Electronics Engineering, and currently works as a Network Engineer at Facebook.

Shikha Suneja is a graduate student from the University of Colorado Boulder with a major in Network Engineering. She received her Bachelor of Engineering in Electrical and Electronics Engineering, master's degree in Mathematics, and currently works as a Production Engineer at Facebook.

Srinidhi Vajapeyam Srivatsa is a graduate student from the University of Colorado Boulder with a major in Network Engineering. He received his Bachelor of Engineering in Telecommunications Engineering, and currently works as an Associate Systems Engineer at Juniper Networks.

Vaishali Ananthasubramanian is a graduate student from the University of Colorado Boulder with a major in Network Engineering. She received her Bachelor of Engineering in Electrical, Electronics and Communications Engineering, and currently works as a Rotational Production Network Engineer at Facebook.

Yogisai Maramraj is a graduate student from the University of Colorado Boulder with a major in Network Engineering. He received his Bachelor of Technology in Electrical, Electronics and Communications Engineering, and currently works as a Systems Development Engineer at Amazon Web Services.

Dr. Levi Perigo is a Scholar in Residence and Professor of Network Engineering in the Department of Computer Science, University of Colorado Boulder. His interests are in a variety of internetworking technologies such as network automation, VoIP, IPv6, SDN/NFV, and next generation protocols. Currently, his research focuses on implementation and best practices for network automation, SDN, and NFV.

Rahil Gandotra is a Ph.D. candidate at the Interdisciplinary Telecom Program, University of Colorado Boulder. He received his bachelor's degree in Telecommunications Engineering and has primary research interests in next-generation networking focusing on software-defined networking, network functions virtualization, and energy-efficient networking.

Dewang Gedia is a Ph.D. candidate at the Interdisciplinary Telecom Program, University of Colorado Boulder. He received his Bachelor of Engineering in Electrical, Electronics and Communications Engineering, master's degree in Network Engineering, and has primary research focus in network functions virtualization and software-defined networking.